# Efficient Homomorphic Integer Polynomial Evaluation based on GSW FHE

Husen Wang, Qiang Tang

University of Luxembourg
husen.wang@uni.lu, qiang.tang@uni.lu

December 14, 2016

**Abstract** In this paper, we introduce new methods to evaluate integer polynomials with GSW FHE. Our solution has much slower noise growth and per homomorphic integer multiplication cost, which is only $O(\frac{(\log k)^{w+1}}{k^w \cdot n})$ percent of the original GSW scheme, where $k$ is the input plaintext width, $n$ is the LWE dimention, and $\omega = 2.373$. Technically, we reduce the integer multiplication noise by performing the evaluation between two kinds of ciphertexts, one in $\mathbb{Z}_q$ and the other in $\mathbb{F}_2^{\lceil \log q \rceil}$. To achieve generality, we propose an integer bootstrapping scheme which converts these two kinds of ciphertexts into each other. To solve the ciphertext expansion problem due to ciphertexts in $\mathbb{F}_2^{\lceil \log q \rceil}$, we propose a solution based on symmetric encryption with stream ciphers.

**Keywords:** GSW, Homomorphic Encryption, integer multiplication, Polynomial, bootstrapping, packing

## 1 Introduction

Fully Homomorphic encryption (FHE) has received considerable attention since the breakthrough by Gentry in [15], which put forward a paradigm for converting a "somewhat homomorphic" encryption scheme with a limited evaluation depth to "fully homomorphic" encryption scheme with unlimited depth, by bootstrapping a noisy ciphertext to less noisy one. Since then, a lot of efforts have been dedicated to make FHE more practical by improving the evaluation performance and reducing the number of bootstrapping times. The second generation of schemes such as BGV [9], LTV [19](NTRU based), its scale-invariant version like FV [14] and YASHE [7] utilize techniques such as bit decomposition, modulus switching, key switching to reduce the noise growth when performing evaluations, particularly multiplications. However, the unnatural key switching leads to serious computation overhead. To resolve this problem, the third generation FHE was proposed by Gentry in [16], which encrypts the plaintexts as eigenvalues of matrix ciphertexts, while the secret key as the eigenvector. Later,

[11] achieved quasi-additive noise growth property by utilizing the asymmetric property of the matrix multiplication. [1] improved the bootstrapping procedure by arithmetically evaluating the decryption circuit and embedding elements in $\mathbb{Z}_q$ into smaller symmetric groups using Chinese Reminder Theorem (CRT). Recently, [18] enabled packing multiple bits by adopting a new structure for the ciphertexts, and further improved the bootstrapping procedure in [1]. However, all the above improvements target at binary plaintexts, since the multiplication noise growth in an integer multiplication chain is exponential with the plaintext size.

Later, Leo et al. [13] proposed FHEW which accelerates bootstrapping with the cyclic group elements $\mathbb{Z}_q$ encoded into the group of roots of unity: $i \mapsto X^i$, where $i$ is a primitive $q$-th root of unity. This method was further improved by Jean et al. [4] who choose to bootstrap multiple bits to allow more types of gates by using membership test. Theoretically they can bootstrap any integer in $\mathbb{Z}_p$, but in an inefficient way, since the membership set is limited with the resolution of $m/p$. Besides, the disadvantage of both methods is that the bootstrap is required after every non-free gate and only limited to boolean operations, which is inefficient to build arithmetic circuits.

In this paper, we consider the problem of evaluating integer polynomials, which can be used to approximate functions such as $e^x, \log(1-x), 1/(1-x)$. We perform arithmetic operations between two kinds of ciphertexts, in $\mathbb{F}_2^\ell$ and $\mathbb{Z}_q$, rather than using boolean circuits or direct multiplication of $\mathbb{Z}_q$ elements.

### 1.1 Our contribution

Firstly, we observe that, for univariate polynomials, the most efficient evaluation method is Horner's Rule [6], which can be adjusted so that for every integer multiplication, one operator can always be the input value. We also realize that in GSW scheme, for the homomorphic multiplication, by selecting a special kind of modulus $q$ and setting one operator in $\mathbb{F}_2^\ell$ and the other in $\mathbb{Z}_q$ where $\ell = \lceil \log q \rceil$, we can achieve very low noise increase. As such, during the polynomial evaluation, we keep the accumulator in $\mathbb{Z}_q$ and perform additions and multiplications with the fresh ciphertext encrypting $\mathbb{F}_2^\ell$. This requires us to encrypt the plaintext after decomposing it into an element in $\mathbb{F}_2^\ell$. This bit-decomposition method is a traditional method for FHE cryptosystems such as key switching [8] and bootstrapping [10], but inherently with different motivations and noise effects. For example, these works decompose the ciphertext rather than the plaintext and their noise comes from scaling noise rather our asymmetric noise.

Secondly, generalize our evaluation method by converting the two kinds of ciphertexts encrypting $\mathbb{F}_2^\ell$ and $\mathbb{Z}_q$ elements separately. The conversion $\mathbf{v} \in \mathbb{F}_2^\ell \mapsto \mu \in \mathbb{Z}_q$ is $\mu =< \mathbf{v}, \mathbf{g} >$, $\mathbf{g} = (2^0, 2^1, \cdots, 2^{\ell-1})$, the conversion $\mu \in \mathbb{Z}_q \mapsto v \in \mathbb{F}_2^\ell$ can be seen as the bit decomposition function. The first conversion can be homomorphically computed with very little added noise. For the second conversion, we propose one algorithm to bootstrap a ciphertext encrypting $\mathbb{Z}_q$ to its bit-decomposed $\mathbb{F}_2^\ell$. Interestingly, this also solves the problem for composition functions and the plaintext space expansion for float point polynomials.

Finally, we extend the method to packing by encrypting the integers diagonally in a matrix. We also propose to solve the ciphertext expansion problem with stream cipher encryption and homomorphic decryption [20].

### 1.2 Organization

The rest of the paper is organized as follows. In Section 2 we recall some mathematical preliminaries such as subgaussian variables and DLWE and the GSW variant from [1]. In Section 3 we discuss the homomorphic operations for integers and polynomials, then analyze their noise growth property. In Section 4 we give our bootstrapping algorithm for an integer. In Section 5 we evaluate our computational cost and make a comparison with other GSW based schemes. In Section 6 we extend the evaluation method to packed GSW. In Section 7, we show a real life scenario and propose an efficient procedure. Finally, we conclude the paper.

## 2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of real numbers by $\mathbb{R}$. Let $\mathbb{G}$ be a set, $\chi$ be some probability distribution, then we use $a \xleftarrow{\mathbf{U}} \mathbb{G}$ to denote that $a$ is chosen from $\mathbb{G}$ uniformly at random, and use $b \xleftarrow{\mathbf{R}} \chi$ to denote that $b$ is chosen along $\chi$. We take all logarithms log to base 2, unless otherwise noted.

We assume that column vectors are represented with bold lower case letters, e.g., $\mathbf{x}$, and the transpose as $\mathbf{x}^t$. We also use $()_{i\in[0,\ell)}$ to represent a vector with the length $\ell$. Let $\otimes$ be the tensor product, $\cdot$ be the multiplications between matrices or scalars. The inner product between two vectors is denoted by $< \mathbf{x}, \mathbf{y} >$. We denote $||x||_2$ as the Euclidean norm or the $\ell_2$ norm, $||x||_\infty$ as the Maximum norm.

Matrices are represented with bold capital letters, e.g., $\mathbf{X}$, and the $i$-th column vector of a matrix is denoted by $\mathbf{x}_i$. For matrices $\mathbf{A} \in \mathbb{Z}^{m\times n_1}$ and $\mathbf{B} \in \mathbb{Z}^{m\times n_2}$, $[\mathbf{A}||\mathbf{B}] \in \mathbb{Z}^{m\times(n1+n2)}$ denotes the column concatenation of $\mathbf{A}$ and $\mathbf{B}$. We denote the $n \times n$ identity matrix with $\mathbf{I}_n$. $\ell = \lceil \log q \rceil$. Let $\mathbf{g} = (2^0, 2^1, 2^2, \dots 2^{\ell-1}) \in \mathbb{Z}_q^\ell$, $\mathbf{G} = \mathbf{g}^t \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n\times n\ell}$. For an integer $x \in \mathbb{Z}_q$, we use $x[i]$ to denote the $i$-bit of $x$ and $(x[i])_{i\in[0,\ell)} \in \mathbb{Z}_2^\ell$ to denote the binary representation of $x$.

### 2.1 Subgaussian

Alperin-Sheriff and Peikert [1] used a randomized function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_q^{n\ell\times m}$ instead of the decomposition procedure to randomize the ciphertexts and more accurately analyze the noise growth property. Here, we take the necessary Claims from these related papers.

**Definition 1 ( [27]).** *A real random variable $X$ (or its distribution) is subgaussian with parameter $r > 0$, if for all $t \in \mathbb{R}$, its (scaled) moment-generating function satisfies $\mathbb{E}[exp(2\pi tX)] \leq exp(\pi r^2 t^2)$.*

The subgaussain random variables have two properties:

- Homogeneity: If the subgaussian variable $X$ has parameter $s$, $cX$ is subgaussian with parameter $cs$.
- Pythagorean additivity: If two independent subgaussian random variables $X_1$ and $X_2$ with parameters $s_1$ and $s_2$ respectively, $X_1 + X_2$ is subgaussian with parameter $\sqrt{s_1^2 + s_2^2}$.

**Claim 1** *For $a \in \mathbb{Z}_q$, there is a randomized, efficiently computable function $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^\ell$ such that $\mathbf{x} \leftarrow \mathbf{g}^{-1}(a)$ is subgaussian with parameter $O(1)$ and always satisfies $< \mathbf{g}, \mathbf{x} >= a$.*

**Claim 2** *For $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, there is a randomized, efficiently computable function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}^{n\ell \times m}$ such that $\mathbf{X} \leftarrow \mathbf{G}^{-1}(\mathbf{A})$ is subgaussian with parameter $O(1)$ and always satisfies $\mathbf{G} \cdot \mathbf{X} = \mathbf{A}$.*

### 2.2 DLWE

The LWE problem by Regev [24] and its decisional version $\mathbf{DLWE}_{n,m,q,\chi}$ are recapped in Definition 2. The reductions from $\mathbf{DLWE}_{n,m,q,\chi}$ to $\mathbf{GapSVP}_{\tilde{O}(n/\alpha)}$ based on quantum algorithm [24] and classical algorithm [23] are illustrated in **Corollary 1**, as we rewrite the **Corollary** from [11]. The $\mathbf{GapSVP}_\gamma$ is assumed to be hard, since the best algorithm requires at least $2^{\tilde{\Omega}(n/\log\gamma)}$ time [25].

**Definition 2 (DLWE).** *For $q = q(n) \in \mathbb{N}$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}$,the (average-case) decision learning with errors problem, denoted $\mathbf{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) $m$ samples chosen according to $\mathbf{A}_{s,\chi}$ (for $s \xleftarrow{\mathbf{U}} \mathbb{Z}_q^n$), from $m$ samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote $\mathbf{DLWE}_{n,m,q,\chi}$ the variant where the adversary gets oracle access to $\mathbf{A}_{s,\chi}$, and is not a-priori bounded in the number of samples.*

**Corollary 1 (DLWE to GapSVP).** *Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$ or a product of co-prime numbers $q = \prod q_i$ such that for all $i$, $q_i = poly(n)$, Let $\alpha \geq \sqrt{n}/q$. If there is an efficient algorithm that solves the (average-case) $\mathbf{DLWE}_{n,m,q,\chi}$ problem, then:*

- *There is an efficient quantum algorithm that solves $\mathbf{GapSVP}_{\tilde{O}(n/\alpha)}$ and $\mathbf{SIVP}_{\tilde{O}(n/\alpha)}$ on any $n$-dimensional lattice.*
- *If in addition $q \geq \tilde{O}(2^{n/2})$, there is an efficient classical algorithm for $\mathbf{GapSVP}_{\tilde{O}(n/\alpha)}$ on any $n$-dimensional lattice.*

### 2.3 Alperin-Sheriff-Peikert GSW Symmetric Encryption Scheme

We first describe the GSW FHE variant in [1], which is identical to original GSW FHE scheme [16] except for some syntactic differences.

$\mathsf{Setup}(\lambda, L)$ : Choose a modulus $q = q(\lambda, L)$, the lattice dimension $n = n(\lambda, L)$. $\ell = \lceil \log q \rceil$. The distribution $\chi$ as subgaussian over $\mathbb{Z}$. Output $params = (n, q, \chi, \ell, \mathbf{G})$.

$\mathsf{KeyGen}(params)$ : sample $\bar{\mathbf{s}} \xleftarrow{\mathbf{R}} \chi^{n-1}$, output secret key $\mathbf{s} = [\bar{\mathbf{s}}||1] \in \mathbb{Z}^n$.

$\mathsf{Enc}(params, \bar{\mathbf{s}}, \mu \in \mathbb{Z}_q)$ : $\bar{\mathbf{C}} \xleftarrow{\mathbf{U}} \mathbb{Z}_q^{(n-1)\times n\ell}$, $\mathbf{e} \xleftarrow{\mathbf{R}} \chi^{n\ell}$. Let $\mathbf{b}^t = [\mathbf{e}^t - \bar{\mathbf{s}}^t\bar{\mathbf{C}}]_q$. Output the ciphertext

$$\mathbf{C} = \begin{pmatrix} \bar{\mathbf{C}} \\ \mathbf{b}^t \end{pmatrix} + \mu\mathbf{G}$$

$\mathsf{Dec}(params, \mathbf{s}, \mathbf{C})$ : For $q = 2^{\ell-1}$, select the last $\ell - 1$ columns of $\mathbf{C}$ as $\mathbf{C}_{(\ell-1)}$. Then $\mathbf{s}^t\mathbf{C}_{(\ell-1)} = \mu \cdot \mathbf{g}^t + \mathbf{e}'$. Recover Least Significant Bit $LSB(\mu)$ from $\mu \cdot 2^{\ell-2} + \mathbf{e}'_{\ell-2}$, then the next-LSB from $(\mu - LSB(\mu)) \cdot 2^{l-3} + \mathbf{e}'_{\ell-3}$, etc.

**Definition 3.** *For a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{n\times n\ell}$, it's designed to encrypt $\mu \in \mathbb{Z}$ under the secret key $\mathbf{s}$, if there is an error vector $\mathbf{e}^t \in \mathbb{Z}^{n\ell}$, and*

$$\mathbf{s}^t\mathbf{C} - \mu \cdot \mathbf{s}^t\mathbf{G} = \mathbf{e}^t \bmod q$$

If $||\mathbf{e}^t||_\infty < q/8$, $\mathsf{Dec}(params, \mathbf{s}, \mathbf{C})$ can correctly output $\mu$. It also works for the general case $q$ ([21] Sec 4.2), for which we need to solve the closest vector problem (CVP), but in polynomial time considering the small dimension of $\ell$. This bound will set a limit to the final ciphertext noise, which can be estimated according to the evaluated function and the noise growth by performing basic homomorphic operations such as addition and multiplication. These basic homomorphic operation noise bound will be given in the section 3.2.

*Homomorphic Addition.* The addition is performed in the same way as in [1][16] [11], we take the subgaussian analysis in [1].

**Lemma 1.** *For two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n\times n\ell}$ which encrypt $\mu_1, \mu_2 \in \mathbb{Z}$ respectively with error vectors $\mathbf{e}_1, \mathbf{e}_2$, the homomorphic addition is:*

$$\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2) = [\mathbf{C}_1 + \mathbf{C}_2]_q \tag{1}$$

*The result has a error vector $\mathbf{e}_1 + \mathbf{e}_2$.*

*Homomorphic Multiplication.* Firstly, we recap the subgaussian analysis method from [1] and later on we will introduce our new evaluation method in Section 3.

**Lemma 2.** *For two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n\times n\ell}$ which encrypt $\mu_1, \mu_2 \in \mathbb{Z}$ respectively with error vectors $\mathbf{e}_1, \mathbf{e}_2$, the homomorphic multiplication is:*

$$\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2) = [\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)]_q. \tag{2}$$

*This is a randomized procedure as shown in the **Claim** 2, because $\mathbf{G}^{-1}$ is randomized. The result has a error vector $\mathbf{e}^t + \mu_1\mathbf{e}_2^t$, where the entries of $\mathbf{e}$ are independent and subgaussian with the parameter $O(||\mathbf{e}_1||)$.*

## 3 Polynonmial Evaluation

In this section, we first introduce our new homomorphic polynomial evaluation method based on a new encryption procedure $\mathsf{BEnc}$. We then investigate the noise growth about homomorphic multiplication in detail. As to homomorphic addition, the noise growth is small and the same as in other schemes.

### 3.1 Bit Encryption Procedure

With respect to the decryption algorithm $\mathsf{Dec}$, we introduce an different procedure $\mathsf{BEnc}$ which basically decomposes the input into a binary vector, which can be seen as an element in $\mathbb{F}_2^\ell$ and separately encrypts every bit in the vector.

$\mathsf{BEnc}(params, \bar{\mathbf{s}}, \mu \in \mathbb{Z}_q)$ : For an integer $\mu \in \mathbb{Z}_q$, decompose it into binary representation $(\mu[i])_{i\in[0,l)} \in \mathbb{F}_2^\ell$, then encrypt every bit respectively to $\mathbf{C}[i] = \mathsf{Enc}(params, \bar{\mathbf{s}}, \mu[i] \in \mathbb{F}_2)$. The final ciphertext is $(\mathbf{C}[i])_{i\in[0,l)}$, together with a weight $2^i$ for every $\mathbf{C}[i]$.

For a known constant $\alpha \in \mathbb{Z}_q$, we define $\mathbf{M}_\alpha = [\alpha\mathbf{G}]_q$.

Since the GSW variant in [1] is IND-CPA secure based on the $\mathbf{DLWE}_{n,m,q,\chi}$ assumption, it is clear that adopting the new encryption procedure $\mathsf{BEnc}$ will keep the same security property.

### 3.2 Homomorphic Multiplication

Now we consider computing the multiplication of two integers $\mu_1, \mu_2 \in \mathbb{Z}_q$.

$$\mu_1 \cdot \mu_2 = \mu_1 \cdot \mu_2[0] \cdot 2^0 + \mu_1 \cdot \mu_2[1] \cdot 2^1 + \ldots + \mu_1 \cdot \mu_2[\ell-1] \cdot 2^{\ell-1} \tag{3}$$

$$= \sum_{i=0}^{\ell-1} \mu_2[i] \cdot \mu_1 \cdot 2^i \tag{4}$$

Where $\mu_2 = (\mu_2[0], \ldots, \mu_2[\ell-1]) \in \mathbb{F}_2^\ell$ is the binary representation of $\mu_2$. Correspondingly, we can use $\mathsf{BEnc}$ to encrypt $\mu_2$ and homomorphically evaluate the above process.

We derive two Corollaries 2 and 3 for a special $q$ to show the noise growth property. Firstly, we show the selection criteria for $q$.

**Proposition 1.** *There exists such $q$ and a procedure $\mathbf{G}^{*-1} : \mathbb{Z}_q^{n\times n\ell} \to \mathbb{Z}^{n\ell\times n\ell}$ that for any $i \in [0,\ell)$, $\mathbf{M}_{2^i} = 2^i \cdot \mathbf{G} \in \mathbb{Z}_q^{n\times n\ell}$, if $\mathbf{X} = \mathbf{G}^{*-1}(\mathbf{M}_{2^i})$, then $\mathbf{G} \cdot \mathbf{X} = \mathbf{M}_{2^i}$. For any subgaussian variable $\mathbf{e} \in \mathbb{Z}^{n\ell}$ with the parameter $O(||\mathbf{e}||)$, $\mathbf{e}_a = \mathbf{e} \cdot \mathbf{X}$, then $\mathbf{e}_a$ has the parameter $O(||\mathbf{e}_a||) \approx O(||\mathbf{e}||)$.*

*Proof.* Here we don't need the procedure to be randomized. It's easy to find such pair of $q$ and $\mathbf{G}^{*-1}$. Take $q = 2^{\ell-1}$ as an example, $\mathbf{G}^{*-1}$ is the bit decomposition for every element in the matrix to a column vector $\mathbb{Z}^{\ell\times 1}$, ie,

$2^3 \rightarrow \underbrace{(0,0,0,1,0,\ldots,0)^t}_{\ell} \in \mathbb{Z}^{\ell \times 1}$. It can be verified that $\mathbf{G} \cdot \mathbf{X} = \mathbf{M}$. Then $\mathbf{X} = \mathbf{G}^{*-1}(2^i \cdot \mathbf{G}) \in \mathbb{Z}^{n\ell \times n\ell}$ has no more than one non-zero element in every column, so for any $\mathbf{e} \in \mathbb{Z}_q^{n\ell}$, $\mathbf{e}_a$ is about a permutation of $\mathbf{e}$ and they have about the same subgaussian parameter. Here we use "$\approx$" if the difference is much smaller than the value $O(||\mathbf{e}||)$.

Actually $q = 2^{\ell-1} - \sum_{t_i} 2^{t_i}$ also works, as long as that the set $\{t_i\}$ is small, $t_i < \ell - 1$. □

Fortunately, the restriction on $q$ will not influence the efficiency or security.

**Corollary 2.** *For such $q$ and $\mathbf{G}^{*-1}$ that satisfies the **Proposition 1**, for two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n \times n\ell}$ which encrypt $\mu_1, \mu_2 \in \mathbb{Z}$ respectively with error vectors $\mathbf{e}_1, \mathbf{e}_2$, and a constant $\mathbf{M}_{2^i} = 2^i \cdot \mathbf{G}$, the homomorphic multiplication is:*

$$\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2, \mathbf{M}_{2^i}) = [([\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)]_q) \cdot \mathbf{G}^{*-1}(\mathbf{M}_{2^i})]_q. \tag{5}$$

*The result has an error vector where the entries are independent and subgaussian with the parameter $O(||(\mathbf{e}_1^t + \mu_1 \mathbf{e}_2^t)||)$.*

*Proof.* This is the homomorphic evaluation process of the Equation (3). The multiplication result $[\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)]_q$ has the noise vector $(\mathbf{e}^t + \mu_1 \mathbf{e}_2^t)$, where the entries of $\mathbf{e}$ are independent and subgaussian with the parameter $O(||\mathbf{e}_1||)$. According to the **Proposition 1**, the multiplication with $\mathbf{G}^{*-1}(\mathbf{M}_{2^i})$ will not add extra noise, so the final noise has about the same subgaussian parameter. □

Now, we use the **Corollary 2** to get the noise growth of the homomorphic evaluation of the Equation (3).

**Corollary 3.** *For such $q$ and $\mathbf{G}^{*-1}$ that satisfies the **Proposition 1**, for a ciphertext $\mathbf{C}_1 \in \mathbb{Z}_q^{n \times n\ell}$ which encrypts $\mu_1 \in \mathbb{Z}_q$ with the error vector $\mathbf{e}_1$, and a vector of ciphertexts $(\mathbf{C}_2[i])_{i \in [\ell]}$ encrypting $(\mu_2[i])_{i \in [\ell]} \in \mathbb{F}_2^\ell$ respectively with the noise vector $(\mathbf{e}_2^t[i])_{i \in [\ell]}$, and a vector $(\mathbf{M}_{2^i})_{i \in [\ell]} = (2^i \cdot \mathbf{G})_{i \in [\ell]}$. The homomorphic evaluation of the Equation (3) is:*

$$\mathbf{C}_1 \boxtimes (\mathbf{C}_2[i])_{i \in [\ell]} = \left[ \sum_{i=0}^{\ell-1} (\mathbf{C}_2[i] \cdot \mathbf{G}^{-1}(\mathbf{C}_1)) \cdot \mathbf{G}^{*-1}(\mathbf{M}_{2^i}) \right]_q. \tag{6}$$

*Here, we define the symbol $\boxtimes$ as the operation of one ciphertext with a vector of ciphertexts which encrypt $\mathbb{F}_2^\ell$. The result has an error vector whose entries are independent and subgaussian with the parameter $O(||\mathbf{e}||)$, where in the worst case,*

$$\mathbf{e}^t = [\underbrace{\mathbf{e}_1^t || \ldots \mathbf{e}_1^t}_{\ell} || \mathbf{e}_2^t[0] || \mathbf{e}_2^t[1] || \ldots || \mathbf{e}_2^t[\ell-1]] \in \mathbb{Z}^{2n\ell^2}.$$

*Proof.* Considering every $(\mathbf{C}_2[i] \cdot \mathbf{G}^{-1}(\mathbf{C}_1)) \cdot \mathbf{G}^{*-1}(\mathbf{M}_{2^i})$, the result has an error vector $(\mathbf{e}^t[i] + \mu_2[i]\mathbf{e}_1^t)$, where the entries of $\mathbf{e}^t[i]$ are fresh independent and subgaussian with the parameter $O(||\mathbf{e}_2[i]||)$. Since all the noise $\mathbf{e}_2[i]$ are independent, the final result has the error vector of the sum of $(\mathbf{e}^t[i] + \mu_2[i]\mathbf{e}_1^t)$ for all $i$. The worse case means $\mu_2[i] = 1$ for all $i$. □

### 3.3 Homomorphic Polynomial Evaluation

For any integer univariate monic polynomial

$$\mathsf{F}(x) = p_0 + p_1 \cdot x + \ldots + x^d \tag{7}$$

$$= p_0 + (p_1 + (p_2 + (\ldots (p_{d-1} + \underbrace{x) \cdot x \ldots) \cdot x) \cdot x}_{d} \tag{8}$$

This Horner's Rule allows one operand in every multiplication to be the fresh input $x$ all the time, so that all the homomorphic multiplication can be performed in the way as mentioned in **Corollary 3**.

**Corollary 4.** *For a polynomial $F(x) = p_0 + p_1 \cdot x + \ldots + x^d$ with all its coefficients $p_i \in \mathbb{Z}_q$, for such $q$ and $\mathbf{G}^{*-1}$ that satisfies **Proposition 1**, an integer $\mu \in [0, 2^k)$, $k < \ell$. The vector of ciphertexts $(\mathbf{C}[i])_{i\in[0,\ell)}$ encrypting $(\mu[i])_{i\in[0,\ell)} \in \mathbb{Z}_2^\ell$ respectively with the noise vector $(\mathbf{e}[i])_{i\in[0,\ell)} \in \mathbb{F}_2^\ell$, and a vector $(\mathbf{M}_{p_j})_{j\in[0,d)}$ encoding the constant coefficient $(p_j)_{j\in[0,d)}$. The homomorphic evaluation of $\mathsf{F}(\mu)$ is:*

$$\mathsf{Eval}((\mathbf{C}[i])_{i\in[0,\ell)}, \mathsf{F}) = \mathbf{M}_{p_0} + (\mathbf{M}_{p_1} + (\ldots (\mathbf{M}_{p_{d-1}} + \underbrace{\mathbf{G} \boxtimes (\mathbf{C}[i])_{i\in[0,\ell)}) \boxtimes (\mathbf{C}[i])_{i\in[0,\ell)}) \ldots) \boxtimes (\mathbf{C}[i])_{i\in[0,\ell)})}_{d}$$

*Here the multiplication $\boxtimes(\mathbf{C}[i])_{i\in[0,\ell)}$ is performed as shown in **Corollary 3**. The result has an error vector whose entries are independent and subgaussian with the parameter $O(||\mathbf{e}'||)$, in the worst case*

$$(\mathbf{e}')^t = [\ \underbrace{\mathbf{e}_p^t || \ldots || \mathbf{e}_p^t}_{(k^{d-1}-1)/(k-1)}\ ],\ \mathbf{e}_p^t = [\mathbf{e}^t[0] || \mathbf{e}^t[1] || \ldots || \mathbf{e}^t[\ell-1]] \in \mathbb{Z}^{n\ell^2}.$$

*Proof.* Considering that the addition with $\mathbf{M}_{p_i}$ has no noise increase, we can see the polynomial as an integer multiplication chain, which is $x^d = \underbrace{x \cdot \ldots \cdot x}_{d}$. For every integer multiplication of one intermediate ciphertext (with the noise vector $\mathbf{e}_1$) with encrypted $(\mu[i])_{i\in[0,\ell)}$(with the noise vector $(\mathbf{e}[i])_{i\in[0,\ell)}$), the noise growth is $\sum_{i=0}^{\ell-1}(\mathbf{e}^t[i] + \mu[i]\mathbf{e}_1^t) \leq \sum_i \mathbf{e}^t[i] + k\mathbf{e}_1^t$, because $\mu < 2^k$. The noise $\mathbf{e}_1$ and $(\mathbf{e}[i])$ are independent of each other because of the randomized procedure $\mathbf{G}^{-1}$, so iteratively we can calculate that the overall noise growth is $\sum_{j=0}^{j<(k^{d-1}-1)/(k-1)} \sum_{i=0}^{\ell-1}(\mathbf{e}^t[i])$. □

### 3.4 More Techniques

**Non-Adjacent Encoding.** From the **Corollary 2**, we know that if $\mu_1$ is nonzero, the noise in the multiplication result will inherit the noise $\mathbf{e}_2$. To avoid this, we can use Non-Adjacent Encoding [17] to the bit-decomposed integer, which is the vector $(\mu_2[i])_{i\in[0,\ell)} \in \mathbb{F}_2^\ell$ in the equation 3. We choose Booth encoding method [5], which eliminates continuous 1 by introducing $-1$, reducing the worst case number of non-zero bits in $\mathbb{F}_2^\ell$ from $\ell$ to $(\lfloor \ell/2 \rfloor + 1)$.

Another important reason we choose Booth encoding is that it can be efficiently homomorphically evaluated using $2\mu \ominus \mu$, where $\ominus$ denotes a bitwise substraction.

**Partial Computation.** The matrix multiplication cost $\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2, \mathbf{M}_{2^i})$ in **Corollary 2** can be reduced to $O(1/n(n\ell)^\omega)$ instead of $O((n\ell)^\omega)$. Considering the fact that to decrypt a ciphertext $\mathbf{C}$, only the last $\ell$ columns are necessary, so for $[\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)]_q$, only the last $\ell$ columns need to be calculated. Considering another fact that $\mathbf{M}_{2^i} = 2^i\mathbf{G}$, $i \in [0, \ell)$ are such matrices that for the last $\ell$ columns, only the last row is non-zero, so for the multiplication with $\mathbf{G}^{*-1}(\mathbf{M}_{2^i})$, only the right most $\ell$ columns are needed. In fact, since $\mathbf{G}^{*-1}(\mathbf{M}_{2^i})$ is quite sparse, the matrix multiplication is equivalent to column permutation and a few additions.

The same optimization can be done for the homomorphic addition operation. The addition with $\mathbf{M}_{p_i}, i \in [0, d)$ can be further accelerated to $O(\ell)$ additions since the non-zeros are only one row in the last $\ell$ columns.

**CRT (Chinese Reminder Theorem).** For polynomial evaluation, the plaintext space may be very large, which means a larger $q$ and more computational cost. We can use CRT with a set of co-prime modulus $\{q_1, \ldots, q_t\}$ and cryptosystems separately based on $q_i$, $i \in [1, t]$. Even though we have some restrictions such as $q_i = 2^{\ell-1} - 2^k - 1$, it's still easy to select enough co-prime modulus.

It is worth noting that in the original GSW scheme [16], the multiplication noise is so influenced by the plaintext size that it's impossible to multiply with one ciphertext that encrypts a plaintext close to $q$, because the resulted noise will be greater than $q/8$ and make it impossible to decode.

**Optimized Storage.** $\mathbf{M}_{p_i}$ and $\mathbf{M}_{2^i}$ are sparse matrices with $n\ell$ non-zero elements each, so there is no need to store them and the calculation can be done online in a very fast way with just shifting and modular operations.

## 4 Bootstrapping an Integer for GSW

One problem about our evaluation is that we actually use a special kind of fresh ciphertext which encrypts a $\mathbb{F}_2^\ell$ element, while all the intermediate ciphertexts

are those encrypting $\mathbb{Z}_q$, so one problem is how to convert between them. Another issue is that to continuously evaluate multiple polynomials, we need to refresh the ciphertext to one that encrypts $\mathbb{F}_2^\ell$, otherwise the composition of these polynomials leads an exponential increase in the polynomial degree.

It's easy to convert $\mathbf{v} = (\mu[i])_{i\in[0,\ell)} \in \mathbb{F}_2^\ell$ to $\mu \in \mathbb{Z}_q$ by homomorphically using $\mu =< \mathbf{v}, \mathbf{g} >$. However, the inversion needs homomorphic bit decomposition algorithm. We notice that when $q$ is a power of two, the decoding algorithm in [21] is actually a LSB to MSB(Most Significant Bit) extraction algorithm, so by homomorphically perform this decoding algorithm, we can achieve bit decomposition. The key reason it works is that the homomorphic multiplication with $\mathbf{M}_{2^i}$ barely increases the noise.

In detail, the bootstrapping algorithm works as described in Algorithm 1. Since the modulus is a power of two, the plaintext has at most $\ell - 1$ bits, so the bootstrapped result has $\ell - 1$ ciphertexts.

**Algorithm 1:** Bootstrapping an integer

**INPUT:** Given $q = 2^{\ell-1}$, the bootstrapping key set $bk$, a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{n\times N}$, which encrypts an integer $\mu \in \mathbb{Z}_q$
**OUTPUT:** $(\mathbf{C}[i])_{i\in[0,\ell-1)}$, which encrypts $(\mu[i])_{i\in[0,\ell-1)} \in \mathbb{F}_2^{\ell-1}$
$i = 0$
**while** $i < \ell - 1$ **do**
    set $cvec$ as the $(N - 1 - i)$-th column of $\mathbf{C}$
    $\mathbf{C}[i] = \mathsf{Bootstrap}(bk, cvec)$ // $\mathbf{C}[i]$ encrypts $\mu[i]$
    $\mathbf{C} = \mathbf{C} - \mathbf{C}[i] \cdot \mathbf{G}^{-1}(\mathbf{M}_{2^i})$
    $i = i + 1$

In this algorithm, we use the bootstrapping method from [1] as the function $\mathsf{Bootstrap}(bk, cvec)$, which generates a bootstrapped ciphertext that has independent subgaussian entries with parameter $O(sn\ell\sqrt{rdq}) = \tilde{O}(sn\ell\lambda)$, except with probability $2^{-\Omega(n\ell)}$, where $s$ is the fresh ciphertext subgaussian parameter and $r$ is the maximal prime-power divisor used in CRT, $d = \tilde{O}(\lambda)$.

The bootstrapping process will be correct, as long as the noise before every bootstrapping satisfies the requirement in [1], which means that the noise of $\mathbf{C} - \sum_{i\in[0,\ell-3]} \mathbf{C}[i] \cdot \mathbf{G}^{-1}(\mathbf{M}_{2^i})$ should be small. Considering all the $\mathbf{C}_i$ are bootstrapped "fresh" ciphertexts, the noise for $\mathbf{C}$ is lower bounded by $O(sn\ell\sqrt{rdq(\ell-2)})$. Therefore, in order to achieve FHE, $q$ need to be about $(\sqrt{\ell - 2} = \sqrt{\lceil \log q \rceil - 2})$ times larger, which is a small factor.

The iteration number in the bootstrapping process can be less than $\ell - 1$, if the integer size can be exposed to the evaluator without security problems. Besides, if we are processing a float point based polynomial, we can reduce the precision after the bootstrapping by removing ciphertexts which encrypt these least significant bits. In this way, it solves the plaintext space problem for high-degree polynomials.

We note that the GSW symmetric encryption can be easily transformed to GSW public key encryption. Besides, since our bootstrapping algorithm needs encrypted secret keys, it relies on the circular security assumption [2].

## 5 Performance

In this section, we analyze the noise increase and computational cost of our evaluation method, using the subgaussian variable analysis in [1]. We compare our computational cost with the method based on the original GSW [16] and these based on Boolean circuits [1][11][13][3]. While performing the analysis, we take the $\ell$ ciphertexts expansion into consideration.

### 5.1 Reduced $q$

In our scheme, suppose that we have the fresh GSW ciphertext with independent subgaussian entries with the parameter $s$, the $\ell_2$ norm is $O(s\sqrt{n\ell})$, except with probability $2^{-\Omega(n\ell)}$. The input is bounded in $k$-bit, for which we can represent with a length $k$ binary vector, with $(\lfloor k/2 \rfloor + 1)$ non-zeros in the worst case by Booth encoding.

The initial ciphertext noise has subgaussian parameter $s\sqrt{\ell \cdot n\ell}$. Based on **Corollary 3**, after one homomorphic multiplication, the noise will have subgaussian parameter $s\sqrt{(k/2 \cdot \ell + \ell) \cdot n\ell}$. By recursively perform this analysis, for a polynomial with $d$ degrees, the final noise will have subgaussian parameter $O(s\ell\sqrt{n(k/2)^d})$. In order to successfully decode the ciphertext, $q/8 > O(s\ell\sqrt{n(k/2)^d})$ should hold so that $\log q = \tilde{O}(d \log k + \log n)$.

For the original GSW scheme [16], for a polynomial with the degree $d$, if we evaluate with a binary tree with the multiplication depth of $h = \log d$, the plaintext will grow exponentially from $2^k$ to $(2^k)^{2^h} = 2^{k2^h} = 2^{kd}$. For a multplication depth $h$, the noise will be $(N + 2^k)(N + 2^{2k}) \dots (N + 2^{(k2^h)})B$. To correctly decode the ciphertext, $q/8 > (N + 2^k)(N + 2^{2k}) \dots (N + 2^{(k2^h)})B$ should hold so that $\log q = \tilde{O}(k2^{h+1}) = \tilde{O}(kd)$. If we take the large $N$ into consideration, the noise increases much faster. Besides, it will be impossible to multiply with a ciphertext encrypting an integer close to $q/8$, because the noise will be larger than $q/8$ to cause decode failure.

Overall, our method reduces $q$ size at least from $\tilde{O}(kd)$ to $\tilde{O}(d \log k + \log n)$, which helps reducing the computational cost, see below.

### 5.2 Computational Cost

For the $\mathsf{BEnc}$, we need $\ell$ times ciphertexts compared with [16], since we encrypt the bit decomposed ciphertext.

For the polynomial evalutation, we define our unit cost being one homomorphic integer multiplication, which is reduced to $\ell$ matrix multiplication (Reduced

to $1/n$ due to partial computation). The cost for one homomorphic integer multiplication is $O((n\ell)^\omega \cdot \ell/n) = O(n^{\omega-1}(d\log k + \log n)^{\omega+1})$, where $\omega = 2.373$ according to the fast matrix multiplication algorithm [28].

Compared with the original GSW $O((n\ell)^\omega) = O((nkd)^\omega)$ [16], ours is about $O(\frac{(\log k)^{\omega+1}}{k^\omega \cdot n})$ times less.

Compared with ours, for the polynomial evaluation based on boolean circuits [1][11][13][3], their costs are too high to be practical, even with the fast bootstrapping from FHEW [13]. For a single full adder, 7 NAND gates are necessary and for 32-bit Adder, 127 AND gates are needed [26]. For every NAND or AND gate, one matrix multiplication and bootstrapping are required. The amortized cost is too high to be practical. Besides, it may not be able to efficiently sequentialize the circuit.

## 6 Polynomial Evaluation for Packed GSW FHE

In this section, we extend our method in Section 3 to the packed GSW scheme, and analyse its efficiency properties.

### 6.1 Packed GSW FHE

We explore the possibility of packing more integers into a ciphertext by modifying the scheme from [18] which only deals with binary messages, and try to allow SIMD operations. First, we recap the scheme from [18]. Note that this scheme is public key encryption, while the previous one is symmetric key.

Let $\lambda$ be the security parameter, $r$ be the number of integers to be packed, $L$ be the depth of evaluated circuit.

$\mathsf{Setup}(\lambda, L, r)$ : Choose a modulus $q = q(\lambda, L)$, lattice dimension $n = n(\lambda, L)$, $\ell = \lceil \log q \rceil$, $N = (n+r)\ell$, $\mathbf{G} = \mathbf{g}^t \otimes \mathbf{I}_{n+r} \in \mathbb{Z}_q^{(n+r)\times(n+r)\ell}$. The distribution $\chi$ is subgaussian over $\mathbb{Z}$. $m = O((n+r)\log q)$. Output $params = (r, m, n, q, N, \chi, \mathbf{G})$.

$\mathsf{KeyGen}(params)$ : $\mathbf{A} \xleftarrow{\mathbf{U}} \mathbb{Z}_q^{n\times m}$, $\mathbf{S}' \xleftarrow{\mathbf{R}} \chi^{r\times n}$, $\mathbf{E} \xleftarrow{\mathbf{R}} \chi^{r\times m}$. Let $\mathbf{S} = [\mathbf{I}|| - \mathbf{S}'] \in \mathbb{Z}_q^{r\times(n+r)}$. Set

$$\mathbf{B} = \begin{pmatrix} \mathbf{S}'\mathbf{A} + \mathbf{E} \\ \mathbf{A} \end{pmatrix} \in \mathbb{Z}_q^{(n+r)\times m}$$

Let $\mathbf{M}_{(i,j)} \in \mathbf{Z}_2^{r\times r} (i, j = 1, \ldots, r)$ be the matrix with 1 in the $(i,j)$-th position and 0 in the other. For all $i, j = 1, \ldots, r$, first sample $\mathbf{R}_{(i,j)} \in \mathbb{Z}_2^{m\times N}$, and set

$$\mathbf{P}_{(i,j)} = \mathbf{B}\mathbf{R}_{(i,j)} + \begin{pmatrix} \mathbf{M}_{(i,j)}\mathbf{S} \\ \mathbf{0} \end{pmatrix} \mathbf{G} \in \mathbb{Z}_q^{(n+r)\times N}$$

Output $pk = (\{\mathbf{P}_{(i,j)}\}_{i,j\in[r]}, \mathbf{B})$ and $sk = \mathbf{S}$.

$\mathsf{Enc}(params, pk, \mathbf{M} \in \mathbb{Z}^{r\times r})$ : $\mathbf{R}_{(i,j)} \xleftarrow{\mathbf{U}} \mathbb{Z}_2^{m\times N}$, output the ciphertext

$$\mathbf{C} = \mathbf{BR} + \sum_{i,j\in[r]:\mathbf{M}[i,j]=1} \mathbf{P}_{(i,j)} \in \mathbb{Z}_q^{(n+r)\times N}$$

where $\mathbf{M}[i,j]$ is the $(i,j)$-th element of $\mathbf{M}$.

$\mathsf{Dec}(params, sk, \mathbf{C})$ : For $q = 2^{\ell-1}$, to decode $\mathbf{M} \in \mathbb{Z}_q^{r\times r}$, we calculate $\mathbf{SC} = \mathbf{MSG} + \mathbf{E}$. For the integer $\mu$ in the position $(i,i)$ of $\mathbf{M}$, select the $i$ row, $i\cdot\ell+1$ to $(i+1)\cdot\ell-1$ columns of $\mathbf{SC}$ as $\mathbf{C}_i$. Then $\mathbf{C}_i = \mu\cdot\mathbf{g}+\mathbf{e}$, where $\mathbf{g} = (1, 2, \ldots, 2^{l-2})$. Recover $LSB(\mu)$ from $\mu\cdot 2^{\ell-2} + \mathbf{e}_{\ell-2}$, then the next-LSB from $(\mu - LSB(\mu))\cdot 2^{l-3} + \mathbf{e}_{\ell-3}$, etc.

The scheme from [18] uses matrix plaintexts, ciphertexts and keys in order to realize the packing, so we put the integers here diagonally to achieve parallel homomorphic multiplications. We introduce a new encryption procedure $\mathsf{PBEnc}$.

$\mathsf{PBEnc}(params, pk, (\mu_j)_{j\in[1,r]} \in \mathbb{Z}_q^r)$ : For $r$ input integers $(\mu_j)_{j\in[1,r]} \in \mathbb{Z}_q^r$, decompose them into binary representations, which include $\ell$ vectors, such that for $i \in [0,\ell)$, the weight is $2^i$ and the corresponding plaintext matrix is $\mathbf{M}[i] = \mathrm{diag}(\mu_1[i], \ldots, \mu_r[i]) \in \mathbb{Z}_2^{r\times r}$, then encrypt the matrix with $\mathbf{C}[i] = \mathsf{Enc}(params, pk, \mathbf{M}[i])$. The final ciphertext is $(\mathbf{C}[i])_{i\in[1,\ell)}$, together with a weight $2^i$ for every $(\mathbf{C}[i])_{i\in[1,\ell)}$.

For a known constant scalar $\alpha \in \mathbb{Z}_q$, we define $\mathbf{M}_\alpha = [\alpha\mathbf{G}]_q$.

### 6.2 Correctness and Security

The correctness analysis is the same as that of the unpacked scheme.

**Definition 4.** *For a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{(n+r)\times N}$, it's designed to encrypt $\mathbf{M}$ with $(\mu_i)_{i\in[1,r]} \in \mathbb{Z}_q^r$ in the diagnal, under the secret key $\mathbf{S}$, if there is an error matrix $\mathbf{E}^t \in \mathbb{Z}^{n\ell}$, and*

$$\mathbf{S}^t\mathbf{C} - \mathbf{MSG} = \mathbf{E} \bmod q$$

This naturally comes from the proof in [18] and from [21], if $||\mathbf{E}||_\infty < q/8$, $\mathsf{Dec}(params, \mathbf{s}, \mathbf{C})$ can correctly output $\mu$.

Since the packed GSW FHE variant in [18] is IND-CPA secure based on the $\mathbf{DLWE}_{n,m,q,\chi}$ assumption, it is clear that adopting the new encryption procedure $\mathsf{PBEnc}$ will keep the same security property.

### 6.3 Homomorphic operations

For homomorphic addition and multiplication, we can easily extend the proof in [18] to the $\mathbf{Z}_q$ field.

**Corollary 5.** *For two ciphertexts* $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{(n+r)\times N}$ *which encrypt* $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{Z}_q^{r\times r}$ *respectively with error matrixes* $\mathbf{E}_1, \mathbf{E}_2$*, the homomorphic addition is:*

$$\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2) = [\mathbf{C}_1 + \mathbf{C}_2]_q. \tag{9}$$

*The homomorphic multiplication is:*

$$\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2) = [\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)]_q \tag{10}$$

*This is a randomized procedure, because* $\mathbf{G}^{-1}$ *is randomized. The result has a error matrix* $\mathbf{E}+\mathbf{M}_1\mathbf{E}_2$*, where the entries of* $\mathbf{E}$ *has in the* $i$*-th row the independent and subgaussian entries with the parameter* $O(||\mathbf{e}_{1,i}||)$*,* $\mathbf{e}_{1,i}$ *is the* $i$*-th row of* $\mathbf{E}_1$*.*

We then consider the packed integer matrix multiplication

$$\begin{aligned}\mathbf{M}_1 \cdot \mathbf{M}_2 &= \mathrm{diag}(\mu_{1,1}, \ldots, \mu_{1,r}) \cdot \mathrm{diag}(\mu_{2,1}, \ldots, \mu_{2,r}) \\ &= \sum_{i=0}^{\ell-1} \mathrm{diag}(\mu_{2,1}[i], \ldots, \mu_{2,r}[i]) \cdot \mathrm{diag}(\mu_{1,1}, \ldots, \mu_{1,r}) \cdot \mathrm{diag}(2^i, \ldots, 2^i)\end{aligned} \tag{11}$$

where $\mu_{2,j} = (\mu_{2,j}[0], \mu_{2,j}[1], \ldots, \mu_{2,j}[\ell-1]) \in \mathbb{Z}_2^\ell$ is the binary representation of the element $\mu_{2,j}$ in the matrix $\mathbf{M}_2$, for $j \in [1, r]$.

In the evaluation, if we always keep the $\mathbf{M}_2$ as a fresh ciphertext vector encrypted with our encryption procedure $\mathsf{PBEnc}$, then the noise growth rate can be calculated as follows. The only difference from **Corollary 3** is that $n$ is changed to $n + r$.

**Corollary 6.** *For such* $q$ *and* $\mathbf{G}^{*-1}$ *that satisfies* ***Proposition 1*** *, for a ciphertexts* $\mathbf{C}_1 \in \mathbb{Z}_q^{(n+r)\times N}$ *which encrypt* $\mathbf{M}_1 \in \mathbb{Z}_q^{r\times r}$ *with the error matrix* $\mathbf{E}_1$*, and a vector of ciphertexts* $(\mathbf{C}_2[i])_{i\in[\ell]}$ *encrypting* $(\mathbf{M}_2[i])_{i\in[\ell]}$ *respectively with the noise matrix* $(\mathbf{E}_2[i])_{i\in[\ell]}$*, and a vector of ciphertexts* $(\mathbf{M}_{2^i})_{i\in[0,\ell)} = (2^i \cdot \mathbf{G})_{i\in[0,\ell)}$*. the homomorphic evaluation of the Equation (11) is:*

$$\mathbf{C}_1 \boxtimes (\mathbf{C}_2[i])_{i\in[\ell]} = [\sum_{i=0}^{\ell-1} (\mathbf{C}_2[i] \cdot \mathbf{G}^{-1}(\mathbf{C}_1)) \cdot \mathbf{G}^{*-1}(\mathbf{M}_{2^i})]_q. \tag{12}$$

*This is a randomized procedure, because* $\mathbf{G}^{-1}$ *is randomized. Let* $\mathbf{e}_{2,j}[i]$*,* $\mathbf{e}_{1,j}$*,* $\mathbf{e}_j$ *be the* $j$*-th row vector of* $\mathbf{E}_2[i]$*,* $\mathbf{E}_1$ *and the final ciphertext noise respectively, then* $\mathbf{e}_j$ *has independent and subgaussian entries with the parameter* $O(||\mathbf{e}_j||)$*, where*

$$\mathbf{e}_j = [\underbrace{\mathbf{e}_{1,j}||\ldots||\mathbf{e}_{1,j}}_{\ell}||\mathbf{e}_{2,j}[0]||\mathbf{e}_{2,j}[1]||\ldots||\mathbf{e}_{2,j}^t[\ell-1]] \in \mathbb{Z}^{2n\ell^2}.$$

The proof can be done similar to **Corollary 3**, while the only difference is that we treat the error in every row in the ciphertext independently. We skip the details here.

For the polynomial evaluation, we can perform noise growth analysis similar to the **Corollary 4**. We can consider every polynomial evaluation as a matrix multiplication chain, which is $\mathbf{M}^d = \underbrace{\mathbf{M} \cdot \ldots \cdot \mathbf{M}}_{d}$. The input is a vector of ciphertexts $(\mathbf{C}[i])_{i\in[\ell]}$ with the noise $(\mathbf{E}[i])_{i\in[\ell]}$. For every multiplication, the noise of the two ciphertexts is independent of each other because of the randomized procedure with $\mathbf{G}^{-1}$, so iteratively we can calculate that the overall noise growth is $\sum_{j=0}^{j<(k^{d-1}-1)/(k-1)} \sum_{i=0}^{\ell-1}(\mathbf{E}[i])$, where all packed integers are bounded with $2^k$. For the $j$-th row in the final ciphertext noise, it has independent subgaussian with parameters $O(||\mathbf{e}'_j||)$, where $\mathbf{e}'_j = \sum_{j=0}^{j<(k^{d-1}-1)/(k-1)} \sum_{i=0}^{\ell-1}(\mathbf{e}_j[i])$, $\mathbf{e}_j[i]$ being the $j$-th row of $\mathbf{E}[i]$.

As to bootstrapping packed integers, we can use a combination of our bootstrapping algorithm in Section 4 and the method in [18]. Basically it's a process of homomorphically extracting one ciphertext that encrypts a single integer and then bootstrap it. We skip the details for simplicity.

## 7 Integration with Ciphertext Compression

In this section, we analyze two scenarios that we may be confronted with based on our proposed methods.

- In one scenario, the inputs are all integers encrypted as elements in $\mathbb{F}_2^\ell$. This is the most convenient scenario for the evaluator, but the communication cost is high and a burden for the client. So we propose that the client encrypts all their integers bitwisely with a stream cipher and the evaluator homomorphically decrypts and performs the polynomial evaluation.
- In the other one, the inputs come from different parties, so some of them may have plaintexts in $\mathbb{F}_2^\ell$ and some in $\mathbb{Z}_q$. We can perform bootstrapping on these encrypted in $\mathbb{Z}_q$ to convert them to $\mathbb{F}_2^\ell$, then perform the polynomial evaluation.

Traditionally, to securely evaluate a specific function with an untrusted server, the user needs to homomorphically encrypt the data, then upload the ciphertest to the server who can then do perform the computations. Usually, the ciphertext size is very large compared with the original plaintext, and it is referred to as the ciphertext expansion problem. To avoid this problem, [22] proposed to send the data encrypted with a block cipher such as AES to the cloud, so that the server can homomorphically decrypt them and perform evaluation afterwards. However, existing symmetric ciphers encrypt the data with non-linear functions and boolean circuits such as XOR, AND, which require the plaintext to be encrypted as binary vectors. This forces the underlying homomorphic scheme to deal with boolean circuits, which is quite inefficient in integer polynomial evaluation.

For symmetric ciphers, the depth of the decryption circuit need to be small enough to allow further homomorphic evaluations. It is because, for somewhat homomorphic encryption, the total depth for a specific parameters set is fixed.

To minimize the overhead of the decryption circuit, different block ciphers and stream ciphers have been investigated, e.g. in [12,20]. The block ciphers such as AES, Simon-32/64, have a lot of rounds to guarantee the security level, which results in a high decryption depth such as 32 for Simon-32/64, 44 for Simon-64/128, 40 for AES-128. The stream ciphers, on the other hand, has increased noise with the number of decrypted ciphertext blocks, since the homomorphic pseudorandom keystream generation will add noise to the encrypted secret key. To tackle this problem, [20] proposed the stream cipher **FLIP** based on filter permutator, which has such property that the non-linear filtering function always acts on the key bits, rather than the previous output of some function, so the noise level of every decrypted ciphertext is constant. Combined with the additive noise property of the multiplicative chain in GSW, the decryption noise can be restricted to a small number. As shown in [20], for 80-bit and 128-bit security level, **FLIP** has a multiplicative depth of 4.

In our solution, we propose to concatenate the stream cipher **FLIP** decryption with the integer polynomial evaluation circuit. The homomorphic symmetric key decryption needs NAND which will restrict the message space to $\mathbb{F}_2$ and maintain small noise increase, while the polynomial evaluation uses $\mathsf{Add}$ and $\mathsf{Mult}$. What's more, we can integrate the batching method. A high-level workflow is as follows.

- On the user side, for plaintext integers $(\mu_1, \mu_2, \ldots, \mu_r)$, decompose every integer into binary representations

  $$(\mu_1[0], \ldots, \mu_1[\ell-1], \mu_2[0], \ldots, \mu_2[\ell-1], \ldots, \mu_r[0], \ldots, \mu_r[\ell-1])$$

  then aggregate bits with the same weight into the same block such as

  $$(\mu_1[0], \mu_2[0], \ldots, \mu_r[0]), (\mu_1[1], \mu_2[1], \ldots, \mu_r[1]), \cdots$$

  For every block, encrypt it with symmetric encryption, send it to the server together with the corresponding weight.
- On the server side, homomorphically decrypt all blocks into ciphertexts (of the FHE), then homomorphically evaluate the integer polynomial and send the encrypted result back to the user.
- After receiving the encrypted result, the user can decrypt it to get the evaluated result.

## 8 Conclusion

In this paper, we have proposed some methods to improve the univariate polynomial evaluation based on GSW FHE scheme. It is an interesting work to further extend these methods to multi-variate polynomials and Ring-GSW. To make a fair comparison with the state-of-art HE schemes such as BGV, YASHE, FV, etc, detailed security/parameters/homomorphic operations analysis remain to be a future work.

Besides, from the implementation point of view, such as GPU and hardware acceleration, it's easy to adopt the GSW LWE structure, because the matrix multiplication can be highly parallelized considering every element is independent of each other. It avoids the complex logic from FFT. And what's more important is that there is even no need for integer multiplication in $\mathbb{F}_q$ because of the randomization or bit decomposition $\mathbf{G}^{-1}$, since it's just addition of selected columns, which can be pipelined/parallelized or accelerated using SIMD instructions in GPU/CPU.

## Acknowledgements

Both authors are supported by a CORE (junior track) grant from the National Research Fund, Luxembourg.

## References

1. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Advances in Cryptology–CRYPTO 2014, pp. 297–314. Springer (2014)
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems, pp. 595–618. Springer (2009)
3. Biasse, J.F., Ruiz, L.: Fhew with efficient multibit bootstrapping. In: Progress in Cryptology–LATINCRYPT 2015, pp. 119–135. Springer (2015)
4. Biasse, J.F., Ruiz, L.: Fhew with efficient multibit bootstrapping. In: International Conference on Cryptology and Information Security in Latin America. pp. 119–135. Springer
5. Booth, A.D.: A signed binary multiplication technique. The Quarterly Journal of Mechanics and Applied Mathematics 4(2), 236–240 (1951)
6. Borwein, P., Erdélyi, T.: Polynomials and polynomial inequalities, vol. 161. Springer Science & Business Media (2012)
7. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Cryptography and Coding, pp. 45–64. Springer (2013)
8. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Cryptography and Coding, pp. 45–64. Springer (2013)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309–325. ACM (2012)
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on Computing 43(2), 831–871 (2014)
11. Brakerski, Z., Vaikuntanathan, V.: Lattice-based fhe as secure as pke. In: Proceedings of the 5th conference on Innovations in theoretical computer science. pp. 1–12. ACM (2014)
12. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In: FSE 2016. p. to appear in (2016)

13. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology–EUROCRYPT 2015, pp. 617–640. Springer (2015)
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012, 144 (2012)
15. Gentry, C.: A fully homomorphic encryption scheme. Thesis, Stanford University (2009)
16. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)
17. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to elliptic curve cryptography. Springer Science & Business Media (2006)
18. Hiromasa, R., Masayuki, A., Okamoto, T.: Packing messages and optimizing bootstrapping in gsw-fhe. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 99(1), 73–82 (2016)
19. Lpez-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 1219–1234. ACM (2012)
20. Maux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards stream ciphers for efficient fhe with low-noise ciphertexts. In: Advances in Cryptology-EUROCRYPT. p. to appear (2016)
21. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. Advances in Cryptology-EUROCRYPT 2012 pp. 700–718 (2012)
22. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 113–124. ACM (2011)
23. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 333–342. ACM (2009)
24. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM 56(6), 1–40 (2009)
25. Schnorr, C.P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical computer science 53(2), 201–224 (1987)
26. Stefan, T., Nigel, S.: `https://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/`
27. Vershynin, R.: Compressed sensing (2012), `www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`
28. Williams, V.V.: Multiplying matrices in o (n2. 373) time. preprint (2014)